Experimental evaluation of the IP address space randomisation (IASR) technique and its disruption to selected network services

Maxwell Dondo
DRDC – Ottawa Research Centre

Defence Research and Development Canada
Scientific Report
DRDC-RDDC-2014-R146
November 2014

Abstract

In recent years, some computer network defence (CND) researchers and experts have been suggesting the use of moving target defence (MTD) as a proactive cyber security approach. MTD is a set of network defence techniques such as randomisation, deception, etc., that significantly increases the attacker's work effort. One randomisation technique, called internet protocol (IP) address space randomisation (IASR), periodically or aperiodically makes random changes to the network's IP addresses. This makes it harder for attackers to achieve their goals. However, despite its security benefits, this defence technique disrupts the functioning of some network services. It is therefore important to understand the level of disruption that comes with the technique.

In this work, we experimentally evaluate IASR and its disruptive effects on selected network services. Using virtual machines (VMs), we carried out this experiment by setting up a typical computer network that supports selected network services, namely ping, mail, web, and streaming video. We transformed a typical zoned computer network into a flat network and implemented IASR on it. Then, we executed the four selected network services during IASR and made observations on how disruptive the technology could be on these services. The results of our experimental evaluation show variations in performance degradation in some of the selected services when hosts' IP addresses are changed during IASR, suggesting the need for IASR-aware services if this technology is to be effectively adopted for CND.

Significance for defence and security

This report was a deliverable for the Advanced Computer Network Operations (CNO) Tools and Techniques (ACTT) project, whose objective was to study advanced CNO tools and techniques for computer network defence. The experimental work to evaluate the internet protocol (IP) address space randomisation (IASR) technique and its disruptive effects on services in an operational network, partially shows what to expect if such technology is considered for network defence. If network planners and defenders consider IASR as a network defence technique, degradation in service performance that comes with it should be taken into consideration.

Résumé

Au cours des dernières années, certains chercheurs et experts de la défense des réseaux informatiques (CDN) ont proposé la défense contre les cibles mobiles (Moving Target Defense - MTD) comme approche de cybersécurité proactive. La MTD consiste en un ensemble de techniques de défense des réseaux (répartition aléatoire, déception, etc.) qui augmente considérablement les efforts à déployer par un attaquant. L'une des techniques, la répartition aléatoire de l'espace d'adressage du protocole Internet (Internet Protocol Address Space Randomization - IASR), change au hasard les adresses IP d'un réseau (de façon périodique ou irrégulière) et diminue ainsi le succès des attaques. Toutefois, en dépit de ses avantages sur le plan de la sécurité, la technique interrompt le fonctionnement de certains services du réseau. C'est pourquoi il est essentiel de comprendre le degré de perturbation associé à cette technique.

Dans ce document, nous avons évalué de façon expérimentale l'IASR et ses effets perturbateurs sur certains services du réseau. Pour mener l'expérimentation, nous avons mis en œuvre un réseau informatique typique à l'aide de machines virtuelles (MV) prenant en charge les services du réseau choisis, notamment le *ping*, les courriels, l'Internet et la vidéo en continu. Nous avons transformé un réseau informatique zoné typique en un réseau horizontal, auquel nous avons appliqué l'IASR. Durant celle-ci, nous avons exécuté quatre services du réseau en particulier et consigné le niveau de perturbation observé. Les résultats de cette évaluation expérimentale montrent des variations dans la dégradation des performances de certains des services choisis au moment du changement d'adresse IP des hôtes durant l'IASR. D'après ces résultats, l'application de l'IASR à la CDN ne serait efficace qu'avec des services compatibles avec cette technologie.

Importance pour la défense et la sécurité

Le rapport constituait un produit livrable dans le cadre du Projet de techniques et d'outils évolués visant les opérations de réseaux informatiques (Advanced Computer Network Operations Tools and Techniques - ACTT), projet qui vise l'étude de techniques et d'outils d'opérations de réseaux informatiques (ORI) évolués aux fins de défense des réseaux informatiques (CDN). Les travaux expérimentaux menés en vue d'évaluer la répartition aléatoire de l'espace d'adressage du protocole Internet (Internet Protocol Address Space Randomization - IASR) et ses effets perturbateurs sur les services dans un réseau opérationnel montrent, en partie, à quoi s'attendre si une telle technologie est envisagée pour la défense de réseau. En effet, si les planificateurs et les défenseurs de réseaux envisagent l'IASR comme technique de défense de réseau, ils devront tenir compte de la dégradation des performances des services associée à cette technologie.

Acknowledgements

I would like to thank Matthew Kellett and Christopher McKenzie for their contributions in running tests and providing valuable input to the document's contents.

This page intentionally left blank.

Table of contents

List of figures

1 Introduction

The purpose of this work is to evaluate the disruptiveness of an advanced computer network security technique called internet protocol (IP) address space randomisation (IASR). Computer networks have been traditionally designed and set up as mostly static, so a technique like IASR can be used to proactively defend them against cyber attacks.

IASR is one of the computer network security techniques that falls under an umbrella of techniques called moving target defence (MTD). The MTD approach uses proactive techniques such as randomisation, deception, camouflage, etc., to reduce the chances of successful computer network attacks by increasing the work effort on the attacker [1,2]. Randomisation, for example, continuously scrambles the network's configuration over time to make it unpredictable to the attacker. Randomisation is the set of techniques to which IASR belongs.

IASR is a proactive network defence technique that continuously randomizes the network's IP addresses to confound attackers [3]. As described in Section 2, during IASR, IP addresses are changed periodically or aperiodically. The address changes give the impression of a confused or dysfunctional network [3]. This false impression may discourage some attackers, or may delay them by requiring that they restart information collection to carry out attacks. By the time the attackers execute their attacks, the target addresses may no longer exist, so the attacks might fail.

In earlier work, Michalski [4] points out that "No security technology that is added to aid the protection of a network, process, or device is inserted without some measurable form of hindrance or degradation...". IASR is not an exception to this statement. Network services that depend on IP addresses to communicate inevitably experience disruption during IP address changes [5]. Given the network security benefits of IASR, it is worthwhile to understand IASR's disruptiveness to computer networks. We therefore undertake to evaluate this disruptiveness in this work. We achieve that undertaking by carrying out a lab experiment to test services under IASR. In the experiment, we observed and recorded disruptions to the services.

To perform the experiment, we simulated a typical operational network using virtual machines (VMs). The VM network was initially set up as a static zoned computer network, which typifies most static operational networks. A network zone groups functionally similar computers together. The server zone, which contains the network's servers, is a typical example. For IASR implementation, we transform this network into a flat network. In the flat network, all hosts are connected to the same router. The flat network allows IP addresses to be scrambled without exposing specific network zones to attackers. We scramble the flat network by calculating IP addresses at each host for a given time period. The details of the experiment are given in Section 3.

During the randomisation, we carry out tests and analyse the disruptiveness of IASR on selected services. We tested the *ping* utility as a representation of a connectionless network service. In the connection-oriented service category, we tested both streaming and non-streaming services, namely email, web and streaming video.

Our experimental results, presented in Section 4, partially confirmed what we expected to see–varying levels of service degradation as a result of IASR. Small emails were not disrupted, while some emails with large attachments experienced significant delays. Streaming video worked, but with significant pauses during IP address changes. Web services worked for some small file transfers, but was completely disrupted for very large files. The connectionless ping utility also experienced partial failures. Our experimental evaluation suggests that the selected services need to be made IASR aware to function properly, otherwise work-arounds need to be put in place.

Finally, we present the conclusions to our work in Section 5. Although this work could be an input to future MTD efforts, we do not have immediate plans for follow-up work in this area.

2 IASR background

As mentioned earlier, IASR proactively defends the network by continuously changing the network's configuration. IASR is different from existing static network defence approaches in that it adds a time complexity to the defence. With existing perimeter defence techniques, an attacker is not usually time-constrained to locate and attack vulnerable hosts on the network. However, using the dynamic addressing provided by IASR, an attacker's window of opportunity is limited to the time it takes to change the address. Outside the randomisation window, attempts to target previously identified hosts using their IP addresses are unlikely to succeed. Details of this dynamic addressing technique, IASR, are presented in this section.

The IASR concept is illustrated in Figure 1. In the figure, the left-hand side illustrates the view of the network before randomisation, while the right-hand side illustrates the randomised network. At a time t, the IP addresses of the hosts are as shown in brackets next to each host on the left. For example, the IP address of host $N1$ is `x.y.1.1`. At time $t+\delta t$, where δt is a small change in time, the new network's IP addresses are as shown on the right. In this case, the IP address of host $N1$ becomes `x.y.1.17`.

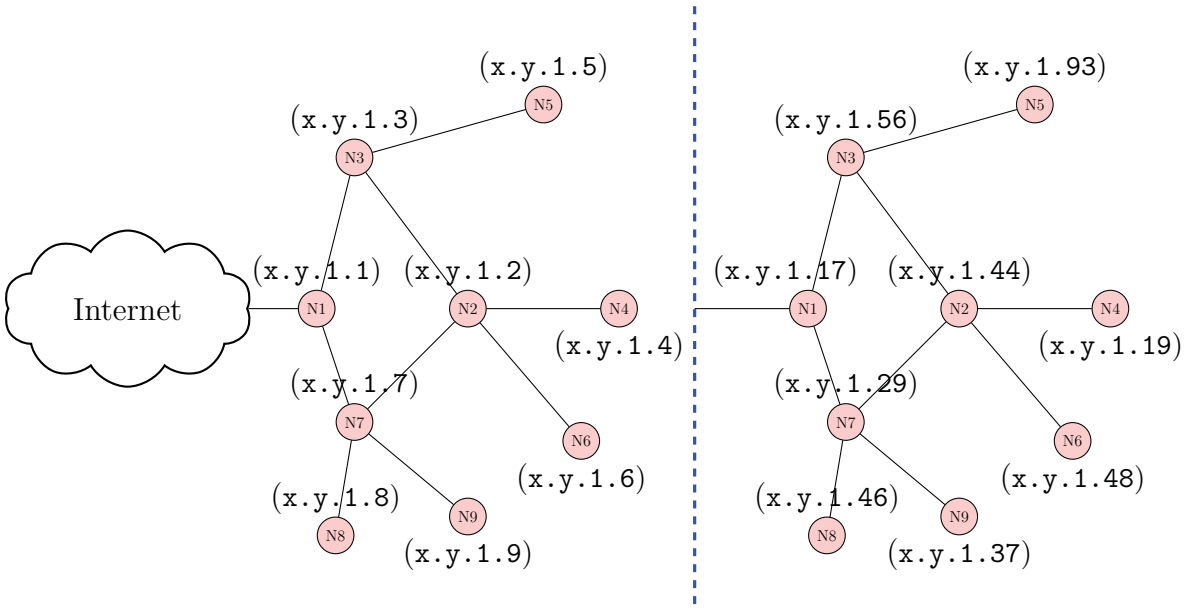


Figure 1: *An illustration of the IASR technique. At time t, the hosts* $Ni : i = 1, \cdots, 9$ *on the left have initial addresses* `x.y.1.i` $: i = 1, \cdots, 9$. *After IASR at time* $t + \delta t$, *the hosts have new addresses as shown on the right.*

The illustrated technique is also called IP hopping, since the hosts hop from one address to another. IP address changes are synchronised in such a way that hosts on

the same network can calculate the other hosts' IP addresses. That allows communications within the randomising network. We use this randomisation technique in our experimental work. For communications outside the randomisation network, the network may require more configuration changes that are beyond the scope of this work.

Methods to determine the dynamic IP addresses depend on the IASR approach used. Examples of IP hopping approaches include mutable networks (MUTE) [6] and network address space randomization (NASR) [7]. MUTE uses round-robin randomisation and NASR uses a lease expiration randomisation. NASR's approach randomises addresses by changing the lease expiration times in the dynamic host configuration protocol (DHCP) at random intervals. When the lease expires, a new address is issued to the host. The approach, however, requires making modifications to the DHCP server in order to force address changes. It also adds extra timing complexities that make randomisation windows of less than 15 minutes difficult if not impossible to implement. In the more flexible MUTE approach, a random IP address is calculated using a crypto-based randomisation function [6]. The use of this function with its secret keys provides unpredictability, while allowing address synchronisation among network hosts. Very low randomisation time windows can be achieved. In addition to these two randomisation examples, other methods of IP hopping are available in other literature such as [3, 5, 8–12].

Our experimental work uses an IP hopping approach similar to MUTE's round-robin randomisation [6]. However, our experiment differs from MUTE's approach in that we do not use a cryptographically secure function. We elected to simplify our experiment by using a simple function that we could easily implement and reverse when necessary. This is a reasonable experimental assumption because the focus of the experiment is on the effects of IP address randomisation on selected services rather than the security of IP address determination. However, for practical implementations of IASR in operational networks, it is essential to use cryptographically secure functions in order to preserve the security of the network.

During IASR, hosts need to be aware of the configuration changes in the network so as to allow for continuous communication. The approach that has been taken by a number of researchers is to use the domain name system (DNS). NASR [7] and MUTE [6] make use of the DNS to serve dynamic IP addresses. Another IASR approach, random host mutation (RHM) [3, 9], which maps hosts to a random virtual IP address, also uses the DNS. Unfortunately, as pointed out by Dunlop *et al.* [13], if the DNS is compromised, all the hosts' dynamic IP addresses would be available to the attacker, thus reducing the security benefits of using IASR.

In their IASR implementation, Dunlop *et al.* [13] avoided using the DNS. Each of their communicating hosts is able to independently calculate their own dynamic IPaddresses.

In our experimental work, we similarly avoided using a central DNS. Each of our hosts is able to independently calculate its own dynamic IP address and those of all the other hosts in the network. However, to carry out tests on the email service, we had to add a decentralised DNS on each communicating host, since email does not work without an assigned DNS. Although this solution is infeasible in large operational networks, that was the only way we could evaluate this service under IASR.

The security benefits of IASR come at a price. Services that communicate through the use of IP addresses are disrupted [4, 5] when the addresses are changed. During regular network communications, a socket is usually established between a client and a server. When the IP address of either communicating host is changed, the socket connection is severed and has to be re-established for communications to resume. This disruption is most pronounced in transmission control protocol (TCP) communications since it is connection-oriented. TCP requires a fully connected session during client-server transactions. If IASR is implemented on an operational network, network security decision-makers must consider its impact on network services. Our experiment, described in the next section, shows the disruptive effects of IASR on selected network services.

3 Experimental setup

In this section, we describe the experimental setup to implement IASR in a lab environment. We first present the setup of a static network representing an operational network setting. Then, we present how we transform this network into a dynamic network that has changing host IP addresses. Our setup allows us to switch between these two network configurations. We also present the use of the DNS server in our experiment.

3.1 Initial zoned network

At the beginning of our experiment, the network is set up to have a zoned static configuration. This setup is typical with most operational networks. The network, which is implemented using VMs, is illustrated in Figure 2.

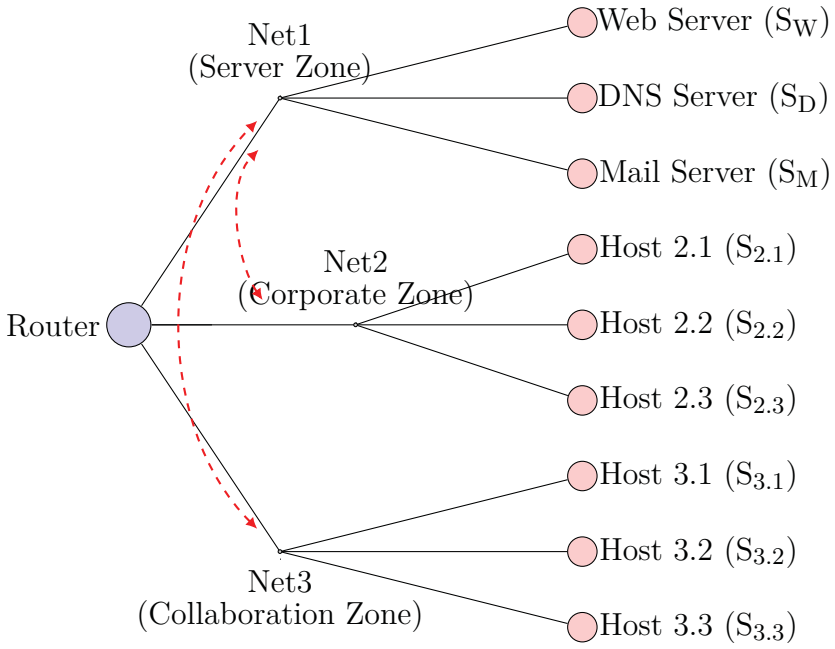


Figure 2: *The experimental network in its static zoned configuration. The static IP address of each host is shown in brackets, e.g. the IP address of the Web server is* S_W. *The dotted arrows represent permitted zone communication.*

The static network shown in Figure 2 is made up of three subnets – Net1, Net2, and Net3. Each subnet is a network zone that has a static network address. Net1 is a Server Zone. Net2 is designated as a Corporate Zone and Net3 as a Collaboration Zone. All subnets are connected to a router using the router's interfaces. The interface connections also serve to implement zone separations. The network implements

additional zone policies through router firewall rules. The policies allow both Net2 and Net3 to communicate with the Server Zone (Net1) and the outside world, but not with each other. The zone communications are illustrated by the dotted arrows in Figure 2.

The experimental network has three hosts on each subnet. The Server Zone has the web, mail and DNS servers. On the other two subnets, we assigned hosts with hostnames that correspond with the subnets they are on. For example, Host 2.1 is on subnet Net2, while Host 3.2 is on subnet Net3. For each host, let S be a static IP address and S_X be the static IP address of host X. Thus, the web server has a static IP address S_W. The rest of the hosts' static addresses are as shown in Figure 2.

3.2 Dynamic flat network

The dynamic network consists of the same network hosts and devices as in the static network. However, we changed the configuration in the dynamic network into one flat network as shown in Figure 3. This configuration change could make it harder for attackers to target a specific zone like the Server Zone.

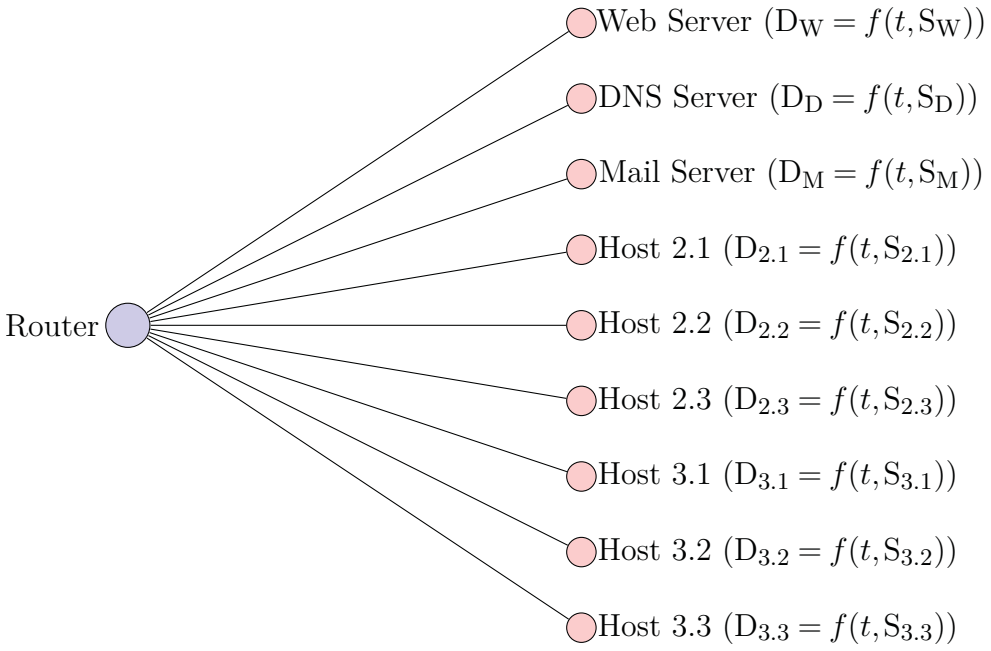


Figure 3: *Logical view of the experimental network with dynamic addressing. All hosts are connected directly to the router. During randomisation, the dynamic IP addresses of each host are time dependent and are as shown in brackets.*

By flattening the network, all hosts are bridged to the router and mingled into one big

subnet with no pattern to distinguish individual hosts. This denies potential attackers any hints as to which zone a host belongs to. In addition, on the flat network, there is a wider range of IP addresses for hosts to be randomised to. As a result, during randomisation, it takes a long time for the IP addresses to cycle back to the same host. This potentially adds an extra degree of difficulty for attackers in finding a valid host IP address on the network.

Once the network is flattened, each host calculates a dynamic IP address. Let D be a dynamic IP address and D_X a dynamic address of host X. That means the web server has a dynamic IP address of D_W. The dynamic IP addresses of all hosts are shown in Figure 3. When IASR is turned off, the network reverts to its original static zoned configuration as shown in Figure 2.

Each host calculates a dynamic IP address D based on the static address S and a time period t. The calculation is shown in Equation 1.

$$D = f(t, S) \tag{1}$$

For our experiment, we used a simple form of Equation 1 that calculates each host's IP address as `10.1.0.yt`, where `t` is a discrete value representing the randomisation time period, such that $t = 1, \cdots, 9$. The duration of `t` (i.e. the time between say $t = 1$ and $t = 2$) is the randomisation window δt, and `y` represents the different hosts, such that $y = 1, \cdots, 9$. For example, the dynamic IP address of the Web Server during randomisation time period 1 is `10.1.0.11`, and that of one Corporate Zone host is `10.1.0.41`. In our experiment, each host calculates the dynamic IP addresses of every other host on the network. The calculation is synchronised to ensure that each host can calculate the dynamic address of all other hosts when needed.

This simple function f makes it easy to revert to static IP addresses. For example, in forensic analysis, it may be necessary to collect network activity logs using hosts' static addresses. In our experiment, this is accomplished by inverting Equation 1 to revert dynamic addresses D back to their static form S. The simple function we use here is not recommended for practical implementations, since attackers could decipher the dynamic IP addresses. As suggested by Al-Shaer *et al.* [6], a strong, cryptographically secure function f that attackers cannot easily decipher should be used instead.

The default randomisation window δt was set to 55 seconds. We think this time duration is short enough to carry out tests, but long enough to avoid significant service interference. Originally, the default randomisation window was set to one minute. However, during the first round of IP address changes, this time setting caused a conflict with a scheduler[1] that ran every minute. We therefore changed δt to 55 seconds to avoid a conflict with the scheduler during the first address change. Subsequent address changes were unaffected by the potential conflict.

[1]The scheduler synchronises randomisation and checks to see if randomisation is still needed.

3.3 Use of the DNS server

As in a typical network, our zoned network makes use of a DNS server. Using the DNS server to serve dynamic addresses is a significant cause of concern [13]. During normal operation, the DNS ordinarily stores all the IP addresses of hosts on the network. However, it is possible that the DNS could get compromised. If that were to happen, then there is a possibility that attackers could have access to all the network's IP addresses, even as they are changed through IASR. For this reason, the network DNS does not serve dynamic addresses, it only serves static addresses. During IASR each host dynamically calculates its and other hosts' IP addresses.

In our experiment, a host-based DNS server was used to serve dynamic addresses in one set of test cases – email experiments. Email requires a DNS server for it to function. We could have used a central DNS server for this experiment. However, to mitigate the potential security risks with the central DNS, we opted for a lightweight host DNS. The host DNS poses less security risks than the central DNS in that it only contains IP addresses for the host and the mail server. A compromise on any of the hosts' DNSs would only reveal the two IP addresses as opposed to revealing all the network's IP addresses as in the case of a central DNS server.

The use of a host DNS in the email experiment is a difficult, if not an infeasible solution for an operational network. Operational networks have thousands of hosts. Managing thousands of DNSs at one time is not very feasible. In addition, the chances of revealing the email server's IP address are higher than when using a central DNS. Each host's DNS could potentially reveal the email server's IP addresses if compromised. However, the attacker would have to compromise all hosts to have access to all of the network's IP addresses. This is a tougher task than getting all the addresses through the compromise of one host, the central DNS server.

With the DNS in place as described, and the network set up for randomisation, IASR is initiated. Selected services are executed and the results recorded as presented in the next section.

4 Experimental results

During the randomisation process, we carried out sets of tests on streaming and non-streaming, connection-oriented and connectionless services. From one host, we executed the connectionless *ping* utility to establish connectivity between hosts during IASR. We also used this utility to test connectivity during subsequent tests on other services. To test web services we performed web-based downloads of files of different sizes from the web server, similar to the approach used by Dunlop *et al.* [5, 13]. On email service experiments, we tested the functionality by sending and receiving emails during IASR. We arbitrarily selected one host and sent email from that host to a registered user on the network. Finally, we tested the use of streaming video during IASR. The Department of National Defence (DND) uses video conferencing among its employees [14], so we chose to test streaming video to show how it could be affected by IASR.

4.1 Time chart

To assist in explaining our experimental results, we use a time chart illustrating the data exchange between communicating hosts. The time charts are for illustration purposes only and are not to scale. As an example of a time chart, we consider two communicating hosts A and B as shown in Figure 4.

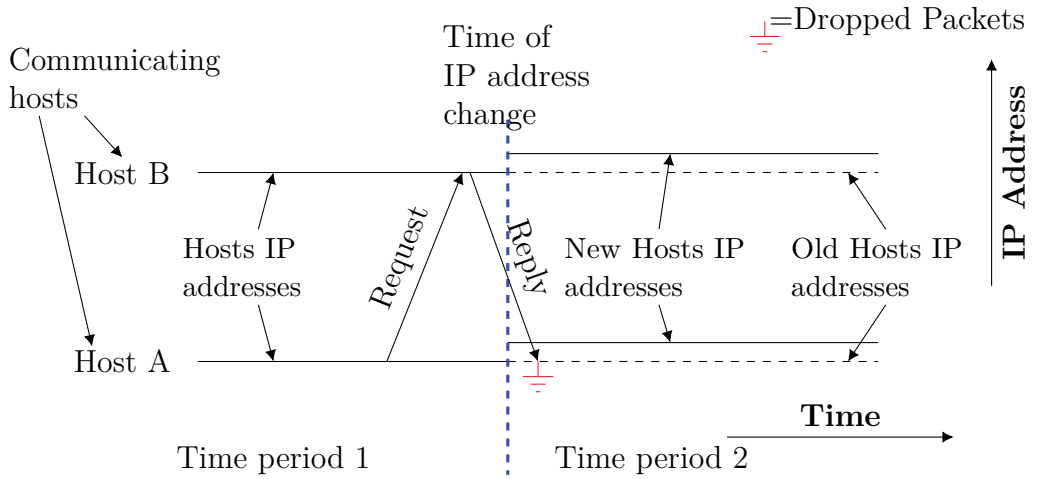


Figure 4: *A time chart illustrating the communications between two hosts, A and B, when the IP addresses of each host are changed between two time periods.*

In Figure 4, the horizontal axis represents the time, and the vertical axis represents the IP addresses. The time axis is divided into time periods of equal duration. Each

time period spans the randomisation time window, which has a default duration of 55 seconds in our experiment. The figure shows two time periods: Time period 1 and Time period 2.

The IP addresses of hosts A and B during Time period 1 are represented by the solid horizontal lines. At some point in time, represented by the dotted vertical line, the IP addresses of both hosts are changed. The new IP addresses for both hosts in Time period 2 are again represented by the solid lines: the top solid line represents the IP address of Host B while the bottom line represents that for Host A. The dotted horizontal lines in Time period 2 represent the IP address for either host in the previous time period (Time period 1 in this case).

In the communication example illustrated in Figure 4, Host A makes a communication request to Host B. The request, which is made in Time period 1, is illustrated by the arrow (arrow with "Request" label) from Host A to Host B. Host B attempts to respond to Host A's request. However, before the response is received by Host A, the hosts' IP addresses are changed. Host B's response (arrow with "Reply" label) is destined to Host A's old IP address. Since this IP address is now invalid, the response fails and results in dropped packets as illustrated. It should be noted that in all the tests that follow, the beginning of communications were not set to coincide with IP address change times or any particular time within the randomisation window; communications were initiated at arbitrarily chosen times.

4.2 Ping

In this section, we report our experimental results on the functionality of the *ping* utility under IASR. The *ping* utility or program sends an internet control message protocol (ICMP) echo request from one host to an identified target, using the target host's IP address [15]. In return, the program expects to receive an ICMP echo reply from the target host. If the target host is configured to respond to ICMP echo request messages, it replies to the request with an ICMP echo reply message using the sender's IP address as identified in the request message. However, while *ping* is running on the requesting host and the target host changes its IP address, the request is lost and never reaches its target. Similarly, if the sender's IP address is changed after sending an echo request, any possible responses from the target host would be lost. The ping program does not perform an address lookup when it does not receive a response from the target. As a result, it would experience failures during IASR.

In our experiment to test the *ping* utility under IASR, we send *ping* messages from Host 3.1 to the mail server (host labeled "Mail Server" in Figure 3)[2]; both hosts accept and respond to ping messages. In the process, we collected different types of

[2]Note that ping can connect to any host on the network, not just the servers as in this case.

logs (tcpdump, syslog, and IASR script logs) to analyse and validate our observations. It should be noted that, although we report on the experiment between Host 3.1 and the mail server, we also ran *ping* experiments between other hosts in the network, and the results were the same as those reported in this section.

When *ping* was executed, it worked for some time (less than 55s) and then failed. We made several other *ping* tests on the same hosts. In every test, *ping* would fail after some time. We noted, as would be expected, that the failure coincided with the IP address change during IASR. We also found out that the failed program would work again if restarted. It would work until the IP address is changed, at which point it would fail again. We explain this *ping* behaviour under IASR using the time chart in Figure 5.

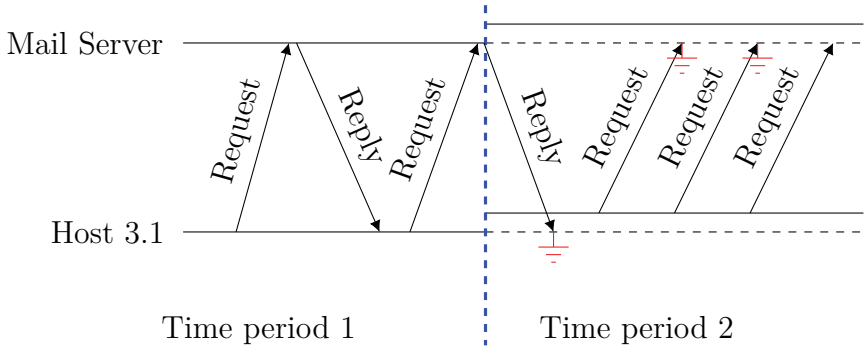


Figure 5: *The failure of the ping utility under IASR. The ping utility executed in period* 1 *fails when the IP address is changed. The utility attempts to reconnect using the IP address from period* 1*, which fails because it is no longer valid.*

When ping is executed on Host 3.1, a request is sent to the the mail server as shown in Figure 5. The mail server responds with a reply to Host 3.1. The request process is repeated, but this time IASR changes the IP addresses for both hosts. The response from the mail server is now lost because the IP address of Host 3.1 is no longer valid. Host 3.1 repeats the request to the mail server using the mail server's Time period 1 IP address, represented by the top horizontal dotted line. This request also fails, since that IP address is no longer valid in Time period 2. All subsequent requests by Host 3.1 would fail since the ping implementation does not perform an address lookup whenever a response is not received from the remote host.

We can therefore conclude that *ping* experiences partial disruption under IASR; it fails when the IP addresses are changed. When *ping* fails under IASR, it does not perform a new lookup, similar to some services we describe later. Restarting it after every failed attempt can be a temporary work-around.

4.3 Non-streaming web services

Non-streaming web services use the hypertext transfer protocol (HTTP) protocol. A client usually uses a program like *wget* or a web browser like Mozilla Firefox to make a connection to the web server. Once the connection to the server is established, the client sends a uniform resource identifier (URI) to the server making a request [16]. The server, if configured to respond to such requests, sends the response back to the client using the IP address in the client's initial request message. If the IP address of the client or server or both are changed between communications, the connection is lost and packets are dropped. Programs like *wget* can perform address lookups when communications fail due to address changes. However, web browsers like Mozilla Firefox depend, by default, on users to refresh the browser every time there is an address change. There is therefore interruption of web services during address changes in IASR.

To evaluate the effects of IASR on web services, we performed an experiment using the command-line HTTP application *wget*. Using the default randomisation time window of 55 seconds, we made a connection from Host 3.1 to the web server (host labeled "Web Server" in Figure 3) to fetch documents of sizes 4kB, 10MB, 20MB, and 3GB. We think that these file sizes, except for the 3GB file, are representative of web transactions in typical operational networks. Starting at arbitrary times, we repeatedly transferred these files one at a time. We performed the transfers over a two minute period and observed the effects through logs and file downloads. The two minute period was long enough to carry out transfer tests.

In our tests, we managed to perform some file transfers without disruption. All successful cases involved the 4kB, 10MB, and 20MB files. However, as explained shortly, there were also transfer failures related to these file sizes. The 3GB file transfer always failed. We identified two ways in which web services are disrupted by IASR. We explain the successful transfer first, and then follow it up with the two failed cases.

A complete successful file transfer is illustrated in Figure 6. Every transfer (for files up to 20MB) was successful and completed in times ranging from 0.2s to 0.6s–times that are all far less than the randomisation window size used. The successful transfer is consistent with the findings of Dunlop *et al.* [5], in which the authors experienced no disruptions during the transfer of small files (in this case less than 10MB).

However, some file transfers that straddled IASR time periods were affected regardless of the file size, which leads us to the second case. In this case, the file transfer was disrupted when IP addresses changed. We noticed the disruption during transfers that straddled time periods as observed in the repeated transfer of small files or the transfer of a 3GB file.

One form of file transfer that straddles a time period is illustrated in Figure 7. Host

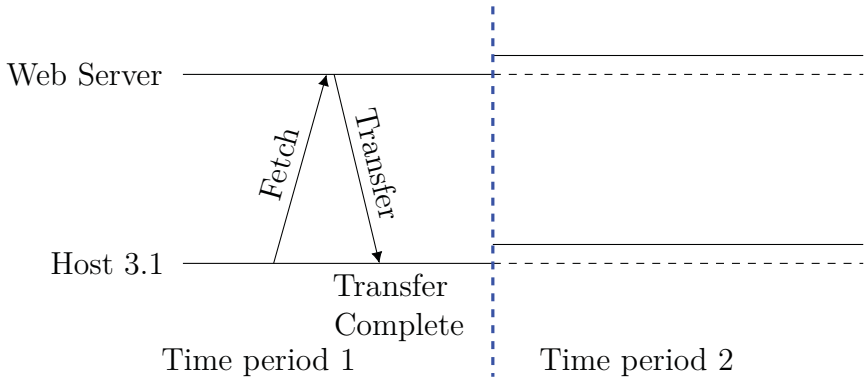


Figure 6: *Successful HTTP transfer within the randomisation window.*

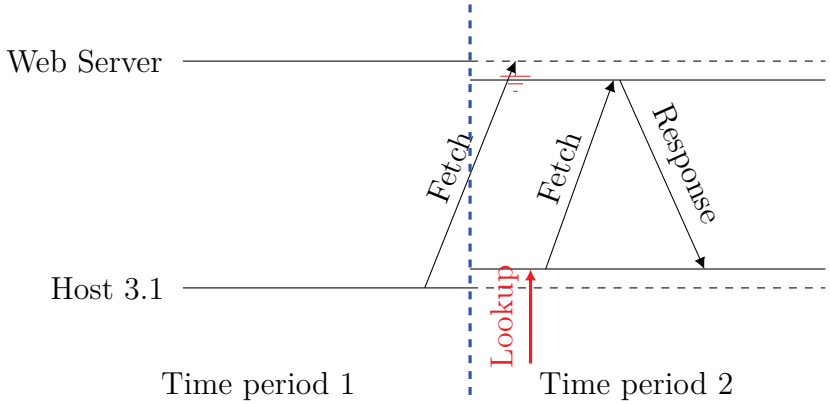


Figure 7: *HTTP transfer just before the IP address change. If the file transfer is interrupted, the host performs an address lookup and communication is restored.*

3.1 attempts to fetch a file from the web server, but fails because the IP addresses of both hosts have changed. Following the failure, Host 3.1 performs an address lookup and gets the correct dynamic IP address. It then completes the transfer using the new IP address provided.

However, depending on when the transfer is initiated, that transfer may completely fail. This failure simply depends on how close the transfer is to the time of IP address change. The application *wget* solved the IP address changes through lookups[3] as explained earlier (see Figure 7) but the transfers would fail at some point.

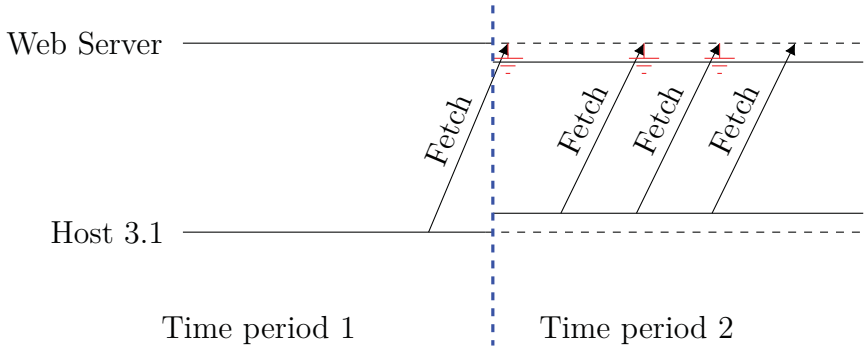


Figure 8: *HTTP transfer overlapping randomisation window. If IP address change interrupts transmission before the completion of the TCP three-way handshake, subsequent communication attempts are made using the IP address for time period* 1*, thus causing the failed transfer.*

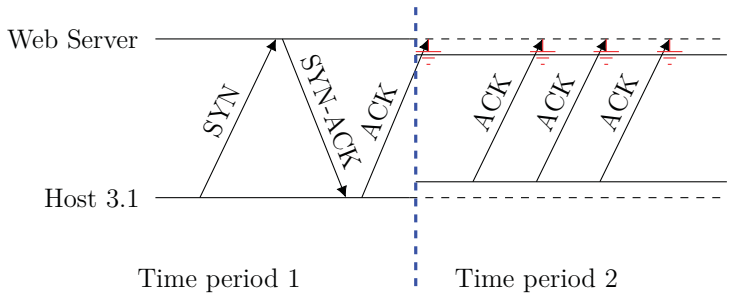
This failure is illustrated in Figure 8. Like the *ping* application, *wget* continues to attempt communications using the destination IP address for the previous time period, resulting in a failure. This continues until *wget* times out and the transfer fails. For small- to medium-sized files, the failure can, like the *ping* utility, be rescued by retrying the transfer. For very large files, retransfer does not always solve the problem since the transfer can still run into a third type of failure and abort the transfer. Through log analysis, we concluded that this failure was due to the interruption of the TCP three-way handshake [15] during IASR.

Figure 9 illustrates the interruption of the TCP handshake during an HTTP file transfer. The first case in Figure 9(a) shows an incomplete connection when the IP address of the server is changed before the ACK packet is received[4]. Figure 9(b) shows an incomplete handshake when the SYN packet is not responded to. In both cases,

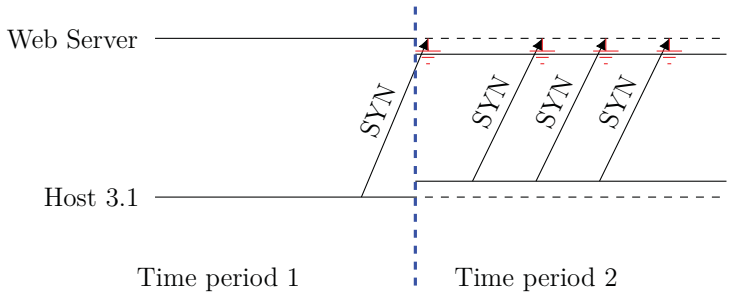
[3]The file transfer was also tested with the Mozilla Firefox browser. The browser would freeze at every IP address change, but would resume from where it left off when "refreshed". Unlike *wget* the browser does not perform a lookup without user intervention.

[4]Similar to the disconnected cable illustration in [15, Chapter 21].

the client (Host 3.1) retries transmission to the old IP address until *wget* aborts with a timeout error.



(a) TCP SYN-ACK failure.



(b) TCP SYN failure.

Figure 9: *Incomplete TCP handshake during HTTP transfer.*

Our tests indicate that for small- to medium-sized files, web services are not significantly affected by IASR. The retry attempts were successfully executed in about 0.3s for all files smaller than 20MB. We were able to repeat these transfers with randomisation windows of 4s, 10s and 20s, with the same results. However, large file transfers were significantly affected by IASR. In fact, *wget* failed to transfer the 3GB file under IASR using the default randomisation window of 55 seconds. In one trial, transfer failed after 2.5 hours for a transaction that would normally take about 75 seconds without IASR. The reason for the complete failure is that the transfer repeatedly ran into the TCP handshake failure illustrated in Figure 9.

A possible work-around for large files can be to increase the randomisation time window from the default 55 seconds, or to change the default settings of *wget*. In fact, we changed the default settings of *wget* to allow for unlimited retry attempts (instead of the default 3) and increased the randomisation time window to 80 seconds, and were able to complete the transfer of all file sizes considered. Whenever the transfer hit the TCP handshake snag described earlier, we manually restarted the transfer.

The functionality of *wget* under IASR can be extended to common web browsers that

are typically used for web services in operational networks. We tested the Mozilla Firefox browser on the Linux operating system (OS), but, as stated earlier, it requires the user to restart it whenever the IP address changes. However, Mozilla Firefox can be made to incorporate *wget* functionalities as add-ons. This makes it capable of providing services under IASR in the same way *wget* does.

4.4 Email

In this section, we report on the experiment to evaluate the effect of IASR email delivery. We consider a message delivery system in which a simple mail transfer protocol (SMTP) client contacts an SMTP server to deliver mail. A user on an SMTP client host uses a mail user agent (MUA) such as Microsoft (MS) Outlook or *pine* to send an email to another user. The MUA, transfers the message to an application called the mail transfer agent (MTA) (such as *sendmail*). The MTA acts as a courier and delivers the message to the email server. Through a series of SMTP handshakes, the MTA sends a request for connection to the server. The server acknowledges the request to allow email transmission[5]. The MTA then sends the email to the server, which stores the emails in recipient mailboxes. The recipient user's MUA receives the emails from the server's mailboxes.

However, if the IP address of the server changes before an acknowledgment is received (or during any part of the SMTP handshakes), the client attempts to resend the email to the IP address it used in the original message. It repeats this attempt three times before performing an address lookup and getting the IP address of the server to which it will eventually deliver the email. The change in IP address results in a delivery delay, which we investigate through this experiment.

For the experiment, we arbitrarily set up one regular 24kB email message and another empty email message with a 20MB attachment. We think that these email sizes represent regular large emails with and without attachments for most operational networks; regular text-based emails are usually smaller than 24kB, and attachments are often capped at 10MB or 20MB. Using the default randomisation window of 55s, we performed the experiment in three stages. In the first stage, we repeatedly transmitted batches of 100 24kB emails. In the second stage, we transmitted batches of 40 emails with 20MB attachments. In the final stage, we created an equal mix of the two email types and transmitted a batch consisting of 40 of each type. We repeated the three stages with window sizes of 20s as well as 2, 4, and 5 minutes. In all cases, we analysed the mail delivery to the user's mailbox on the mail server by using logs produced by our IASR script, *tcpdump*, as well as those produced by the email program, *sendmail*.

[5] A detailed description of how the SMTP protocol works can be found in [15].

From all time windows tested, we observed that the majority of emails were delivered without any ill effects, while a few emails were significantly delayed. In all tests, all the 24kB emails were delivered without delay. However, some emails with the 20MB file attachment experienced delays of up to two hours. In one test, using the default 55s window, 36 out of 40 emails (each with attachments) were delivered without problems. The other four emails were delayed by up to two hours. The first scenario representing the 36 successful email transmissions is illustrated in Figure 10. Similar to HTTP transfer, the emails are delivered within a randomisation window–Time period 1 in this illustrated example.

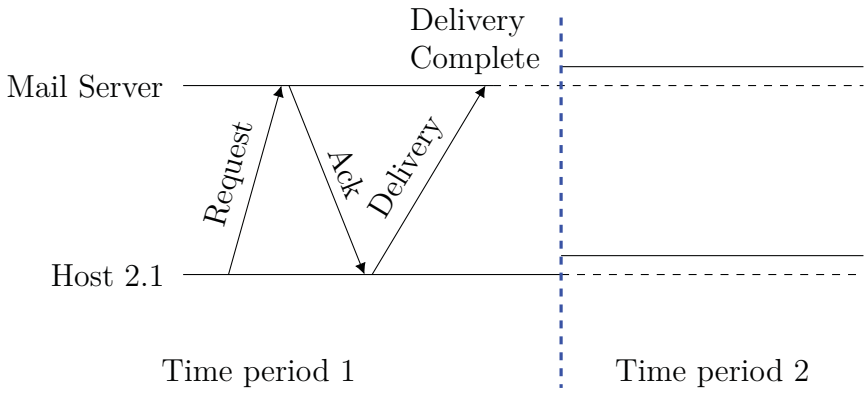


Figure 10: *An email is delivered within the randomisation window.*

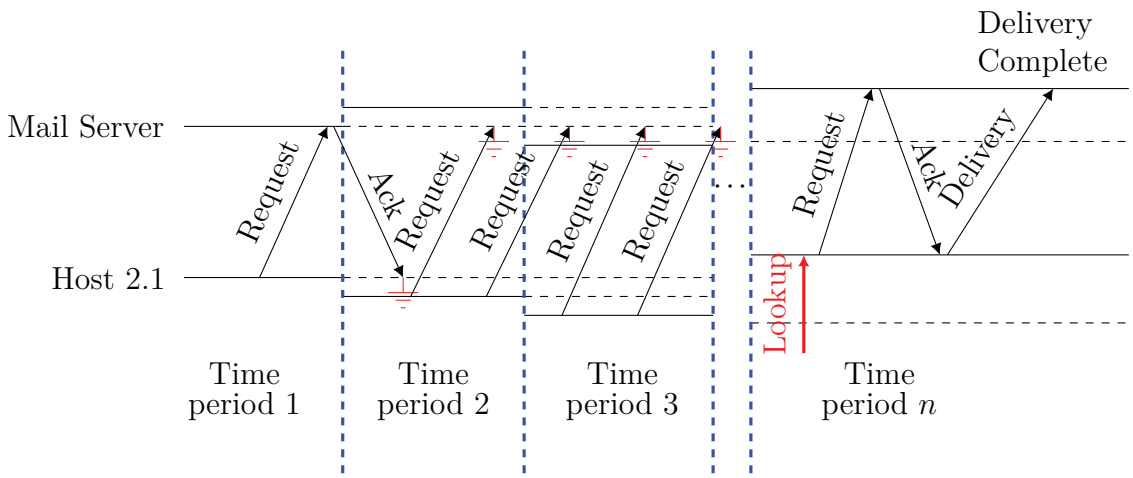


Figure 11: *Email delivery was disrupted, and the application repeatedly tried to reconnect using the IP address from period* 1*, which failed. Delivery was finally successful when the application resolved the new IP address.*

The second scenario representing the four delayed emails is illustrated in Figure 11. The email delivery was affected by IP address change. When a send request was

transmitted, an acknowledgement for final delivery was never received because the IP addresses had changed. Host 2.1 sent another request using the IP address for Time period 1. Since this IP address was no longer valid, no acknowledgement was received. Host 2.1 kept sending requests to the Time period 1 IP address, and of course received no response. Finally, in Time period $n = 126$, Host 2.1 performed a lookup and got the correct IP address, which it used for email delivery.

The email was finally delivered after about two hours. This is not unexpected since the time delay is within the default settings of *sendmail*, which can defer delivery for up to four hours (or a delay set by the administrator) in the event of a delivery failure. After this time, if the email is still undelivered, it is deemed a delivery failure, and a message is sent to the sender advising on the failure. In our experiment, we did not have complete delivery failures; all outstanding emails were delivered following the two-hour delay.

Our tests with other time windows yielded similar results. However, there was a significantly higher number of 20MB emails delayed during tests with a window size of 20s than there was using the default 55s setting. In one such test, 11 of the 40 emails with attachments were delayed (the 24kB emails were not affected at all). Attempts to deliver email at a very low randomisation window of 2s resulted in complete failure. On the other hand, there were significantly lower numbers of delayed emails during tests with randomisation time windows of 2, 4, and 5 minutes. In most cases, all batches of emails were delivered without delays. This behaviour is not unexpected since more emails could be sent within the wider randomisation window than within a narrower one. The few delayed emails are attributed to the few email transmissions that ended up straddling the randomisation time window during transfer.

For email delivery during IASR, there are a number of work-around possibilities. An obvious work-around is to use small emails. Emails of up to 24kB are large enough to cover most general communications in operational networks and we have shown that they are generally unaffected by IASR. Emails with large attachments (up to 20MB), could be retransmitted whenever a delay notification is received. Although retransmission does not guarantee delivery, it increases the chance of the email to be transmitted without being affected by IASR. Another work-around, which may increase network traffic, is to transmit multiple copies of the same large emails (with attachments). As our experiment showed, there is a good chance that at least one of the emails reaches its destination without delay. A final work-around is to use web applications, such as *wget*, to transfer large email attachments. As explained earlier, web services can successfully transfer files of sizes up to 20MB without failures under IASR.

4.5 Streaming video

During video streaming, a client requests multimedia content from the streaming server. The server responds and delivers a data stream to the IP address used in the request. When the IP address of the server changes (such as during IASR), the client detects the change and performs an address lookup and continues to request the multimedia data. This address change results in data loss.

In this section we report on our experiment to evaluate the effects of IASR on streaming video. We set up video streaming between Host 3.1 and the web server (host Web Server in Figure 3). Using the default time window of 55 seconds, we started the video stream while monitoring *tcpdump* and IASR logs. We also carried out tests with other randomisation time windows of 2, 4 and 5 minutes. Our tests were conducted using the real time streaming protocol (RTSP), which is the same protocol that the DND uses for video conferencing [14].

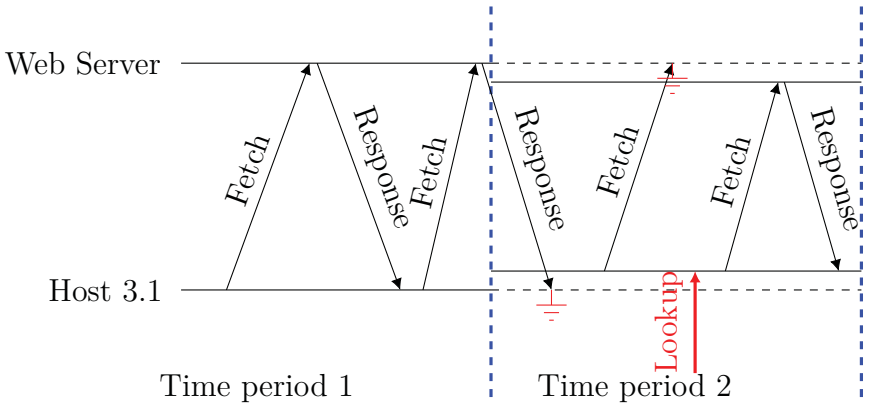


Figure 12: *Video streaming in which video data is transferred over multiple time periods. The application resolves the new IP address every time after the connection is severed.*

Figure 12 illustrates the communications between Host 3.1 and the web server while streaming a video. The video was streamed on the web server and Host 3.1 was connected to it to remotely play the video. From the illustration in Figure 12, Host 3.1 requests the video resource and receives a response. It sends another request, but this time the IP address changes before the response is received. The response is therefore never received. Host 3.1 sends another request using the Time period 1 IP address. Since this IP address does not exist anymore on the remote web server, the request fails. When Host 3.1 does not receive a response, it performs a lookup and gets the current IP address, which it then uses to communicate again and streaming continues. Through observation of the video and analysis of *tcpdump* and IASR script logs the disruption took about 10 seconds. This value was not affected by the size of the randomisation windows that we tested.

The streaming video disruption in our experiment has some similarities with that reported by Dunlop *et al.* [5]. In that work, the authors, who used IPv6 for their testing, experienced significantly low rates of 4 one-second disruptions over a period of about 597s. However, the authors recorded unquantified packet losses during address changes. Their paper seems to suggest that, although there were packet losses during address change, the losses did not affect the video stream. In contrast, our packet losses were significant enough to cause a 10-second disruption in service.

As a work-around, IASR-aware video streaming applications would need to be developed. With the current state of the art, the only way we are aware of that could reduce the disruption to content is to increase the randomisation time window. That way, a 10-second loss in content over 30 minutes or one hour is better than a similar loss every 55 seconds. With randomisation windows as large as 30 minutes or 2 hours, video conferencing may only experience one 10-second disruption for its entire duration.

5 Conclusions

In this work, we have experimentally shown the implementation of internet protocol (IP) address space randomisation (IASR) in a lab environment. In the experiment, we were able to periodically change the networks' IP addresses with time, which is the characteristic of the IASR network defence technique. We have also shown that when IP addresses are changed during IASR, some network communications are either completely or partially lost.

During the communication loss, some network services experience different levels of disruption. For example, using the connectionless *ping* under IASR resulted in failure whenever the communicating hosts' IP addresses changed. The *ping* utility does not perform an address lookup when the IP addresses change, and therefore fails to communicate thereafter. On the other hand, web services experienced less disruption than *ping*. Performing an address lookup after an IP address change ensured that most web services using small files (less than 20MB) were not affected by IASR; there were very few transfer failures for files of this size. However, transmission of very large files (3GB) through web services resulted in prolonged downloads that often resulted in failures. Similar to web services, all small emails that we tested (up to 24kB) were unaffected by address changes. However, some emails with large attachments (20MB) experienced 2-hour delays. Streaming video, which is representative of video conferencing as used by large organisations, showed a 10-second disruption during IASR. Thus, the services we tested in this work cannot be used under IASR without experiencing some level of service degradation at all times.

If IASR were to be implemented in an operational network, then IASR-aware applications or work-arounds be put in place in order to avoid or minimise performance degradation in the services we tested. One possible work-around that we tested in our work is the restarting of services whenever there was a failure due to IP address change. For example, the *ping* utility and web services worked when restarted after an IASR-related failure. Another work-around is to send duplicate emails at the same time; this increases the chances of email delivery without disruption. Unfortunately, this option may not be attractive since it increases network traffic and eventually results in some duplicate emails in mailboxes.

Another work-around is to increase the duration of the randomisation intervals. With longer randomisation windows, disruptions occur less often. For example randomisation windows of 5 minutes or 2 hours (as opposed to our default 55 seconds), would be sufficient to transfer all emails and web files we tested in this work with minimal or no disruption. Whenever failures occur, communications can be restored by restarting the affected service.

Acronyms and abbreviations

ACTT Advanced Computer Network Operations (CNO) Tools and Techniques

CND computer network defence

CNO computer network operations

DND Department of National Defence

DHCP dynamic host configuration protocol

DNS domain name system

HTTP hypertext transfer protocol

IASR IP address space randomisation

ICMP internet control message protocol

IP internet protocol

MS Microsoft

MTD moving target defence

MUTE mutable networks

NASR network address space randomization

MTA mail transfer agent

MUA mail user agent

MV machines virtuelles

OS operating system

RHM random host mutation

RTSP real time streaming protocol

SMTP simple mail transfer protocol

TCP transmission control protocol

URI uniform resource identifier

VM virtual machine

References

[1] National Cyber Leap Year Summit 2009 (2009), National cyber leap year summit 2009 co-chairs report. http://www.cyber.st.dhs.gov/docs/National_Cyber_Leap_Year_Summit_2009_Co-Chairs_Report.pdf. (Access Date : 4 June 2013).

[2] National Cyber Leap Year Summit 2009 (2009), National cyber leap year summit 2009 participantsŠ ideas report: Exploring paths to new cyber security paradigms. https://www.qinetiq-na.com/wp-content/uploads/2011/12/National_Cyber_Leap_Year_Summit_2009_Participants_Ideas_Report.pdf. (Access Date : 4 June 2013).

[3] Jafarian, J. H., Al-Shaer, E., and Duan, Q. (2012), Openflow random host mutation: Transparent moving target defense using software defined networking, In *HosSDN 2012: Proceedings of the first workshop on Hot topics in Software Defined Networks*, pp. 127–132.

[4] Michalski, J. T. (2006), Network security mechanisms utilising network address translation, *International journal of critical infrastructures*, 2(1), 10–49.

[5] Dunlop, M., Groat, S., Marchany, R., and Tront, J. (2011), Implementing an IPv6 moving target defense on a live network, In *Moving Target Research Symposium*.

[6] Al-Shaer, E. (2011), Toward network configuration randomization for moving target defense, In *Moving Target Defense*, Vol. 54 of *Advances in Information Security*, pp. 153–159, Springer.

[7] Antonatos, S., Akritidis, P., Markatos, E., and Anagnostakis, K. (2007), Defending against hitlist worms using network address space randomization, *Computer Networks*, 51(12), 3471–3490.

[8] Michalski, J., Price, C., Stanton, E., Lee, E., Chua, K. S., Wong, Y. H., and Tan, C. P. (2002), Network security mechanisms utilizing dynamic network address translation, (Technical Report SAND2002-3613) Sandia National Laboratories.

[9] Al-Shaer, E., Duan, Q., and Jafarian, J. H. (2013), Random host mutation for moving target defense, In *Security and Privacy in Communication Networks*, Vol. 106 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 310–327, Springer.

[10] Shi, L., Jia, C., Lü, S., and Liu, Z. (2007), Port and address hopping for active cyber-defense, In *PAISI 2007: Proceedings of the 2007 Pacific Asia conference on Intelligence and security informatics*, pp. 295–300, Springer.

[11] Shi, L., Jia, C., and Lu, S. (2008), Full service hopping for proactive cyber-defense, In *ICNSC 2008: IEEE International Conference on Networking, Sensing and Control*, pp. 1337–1342.

[12] Northrop Grumman Systems (2011), ARCSYNE for Mission Aware Cyber Command and Control (AFMACC2). http://www.spacewarfare.org/2011/Briefings-2011/Cyber%20Innovation%20Panel%20-%20Mr.%20Godwin.pdf. (Access Date : 4 June 2013).

[13] Dunlop, M., Groat, S., Urbanski, W., Marchany, R., and Tront, J. (2011), MT6D: A moving target IPv6 defense, In *MILCOM 2011: Military Communications Conference*, pp. 1321–1326.

[14] Department of National Defence (2012), Defence Video Communications System. http://vc.mil.ca/CMA-D/default.htm. (Access Date : 29 January 2014).

[15] Stevens, W. R. (1995), TCP/IP Illustrated: Volume 1: The Protocols, Addison-Wesley Professional.

[16] Thomas, S. (2001), HTTP Essentials: Protocols for Secure, Scalable Web Sites, John Wiley & Sons.

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security markings for the title, abstract and indexing annotation must be entered when the document is Classified or Designated.)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

DRDC – Ottawa Research Centre
3701 Carling Avenue, Ottawa ON K1A 0Z4, Canada

2a. SECURITY MARKING (Overall security marking of the document, including supplemental markings if applicable.)

UNCLASSIFIED

2b. CONTROLLED GOODS

(NON-CONTROLLED GOODS)
DMC A
REVIEW: GCEC APRIL 2011

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Experimental evaluation of the IP address space randomisation (IASR) technique and its disruption to selected network services

4. AUTHORS (Last name, followed by initials – ranks, titles, etc. not to be used.)

Dondo, M.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total containing information. Include Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
November 2014	36	16

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Scientific Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

DRDC – Ottawa Research Centre
3701 Carling Avenue, Ottawa ON K1A 0Z4, Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
05bo	

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC-RDDC-2014-R146	

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

(X) Unlimited distribution
( ) Defence departments and defence contractors; further distribution only as approved
( ) Defence departments and Canadian defence contractors; further distribution only as approved
( ) Government departments and agencies; further distribution only as approved
( ) Defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11)) is possible, a wider announcement audience may be selected.)

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

In recent years, some CND researchers and experts have been suggesting the use of MTD as a proactive cyber security approach. MTD is a set of network defence techniques such as randomisation, deception, etc., that significantly increases the attacker's work effort. One randomisation technique, called IASR, periodically or aperiodically makes random changes to the network's IP addresses. This makes it harder for attackers to achieve their goals. However, despite its security benefits, this defence technique disrupts the functioning of some network services. It is therefore important to understand the level of disruption that comes with the technique.

In this work, we experimentally evaluate IASR and its disruptive effects on selected network services. Using VMs, we carried out this experiment by setting up a typical computer network that supports selected network services, namely ping, mail, web, and streaming video. We transformed a typical zoned computer network into a flat network and implemented IASR on it. Then, we executed the four selected network services during IASR and made observations on how disruptive the technology could be on these services. The results of our experimental evaluation show variations in performance degradation in some of the selected services when hosts' IP addresses are changed during IASR, suggesting the need for IASR-aware services if this technology is to be effectively adopted for CND.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

cyber security; computer security; moving target; IP address randomisation

DRDC | RDDC

SCIENCE, TECHNOLOGY AND KNOWLEDGE FOR CANADA'S DEFENCE AND SECURITY | SCIENCE, TECHNOLOGIE ET SAVOIR POUR LA DÉFENSE ET LA SÉCURITÉ DU CANADA

www.drdc-rddc.gc.ca